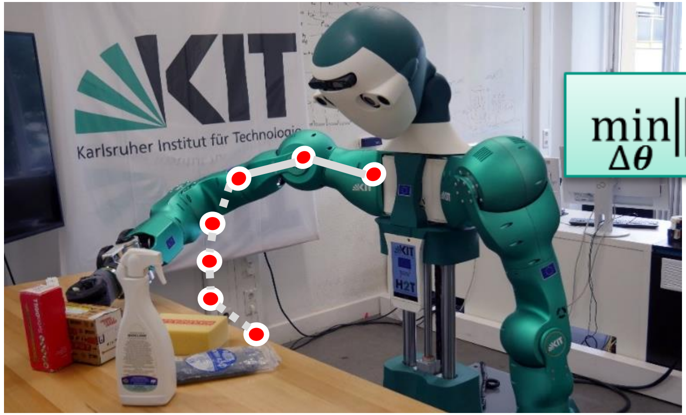
Karlsruhe Institute of Technology

High Performance Humanoid Technologies

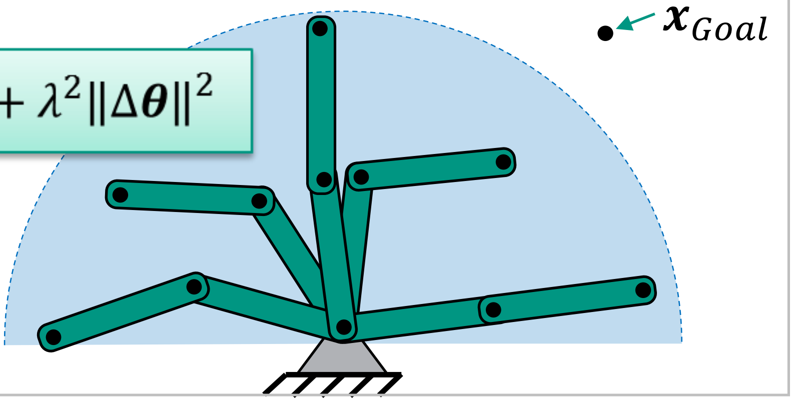
Robotics I: Introduction to Robotics
Chapter 3 – Inverse Kinematics

Tamim Asfour

https://www.humanoids.kit.edu



$$\min_{\Delta\boldsymbol{\theta}} \left\| J_f(\boldsymbol{\theta})\Delta\boldsymbol{\theta} - \Delta\boldsymbol{x} \right\|^2 + \lambda^2 \|\Delta\boldsymbol{\theta}\|^2$$



Overview

- Inverse kinematic problem

- Closed-form methods
  - Geometric
  - Algebraic

- Numerical methods
  - Gradient descent
  - Jacobian based and pseudoinverse based methods

- Summary

Forward Kinematics

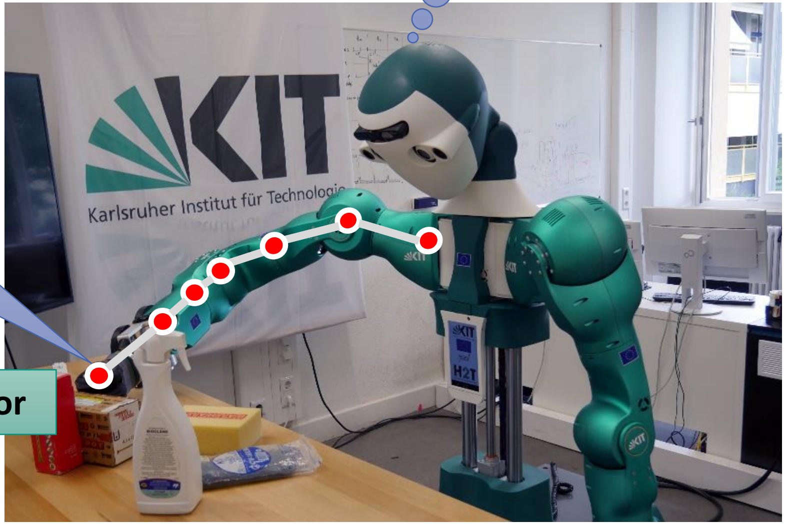
Direct kinematic problem

- Input: Joint angle positions of the robot
- Output: **Pose of the end effector**

Where is my Hand?

Direct kinematics: HERE!

end effector



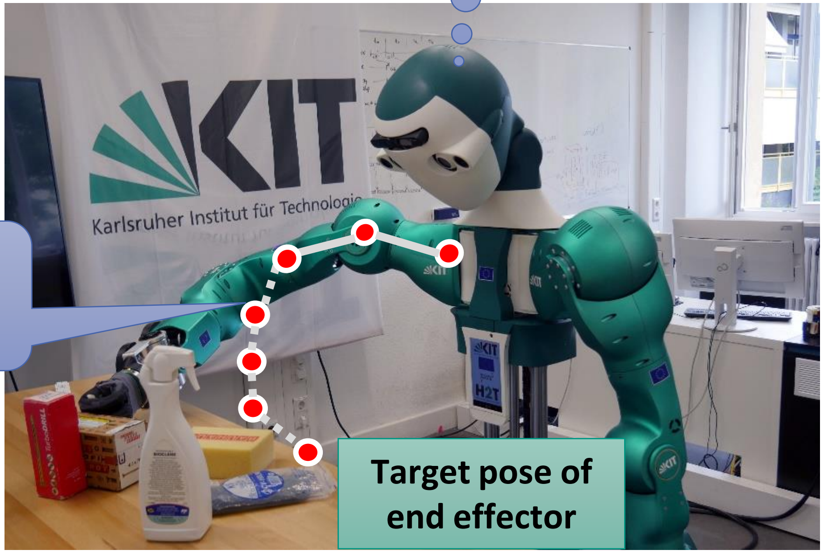
Inverse Kinematics

Inverse kinematic problem:

- Input: Target pose of the end effector
- Output: **Joint angle positions**

How do I move my hand to the target?

Inverse kinematics: Determines the joint angles



Inverse Kinematics

Overview: Direct and Inverse Kinematics

Joint angle
(configuration space)

Transformation

Cartesian coordinates
(task space)

Direct Kinematics

$\boldsymbol{x} = f(\boldsymbol{\theta})$

$(\theta_1, \dots, \theta_n) \in \boldsymbol{C} \subseteq \mathbb{R}^{\boldsymbol{n}}$

$\boldsymbol{x} \in W = \mathbb{R}^{\boldsymbol{m}}$

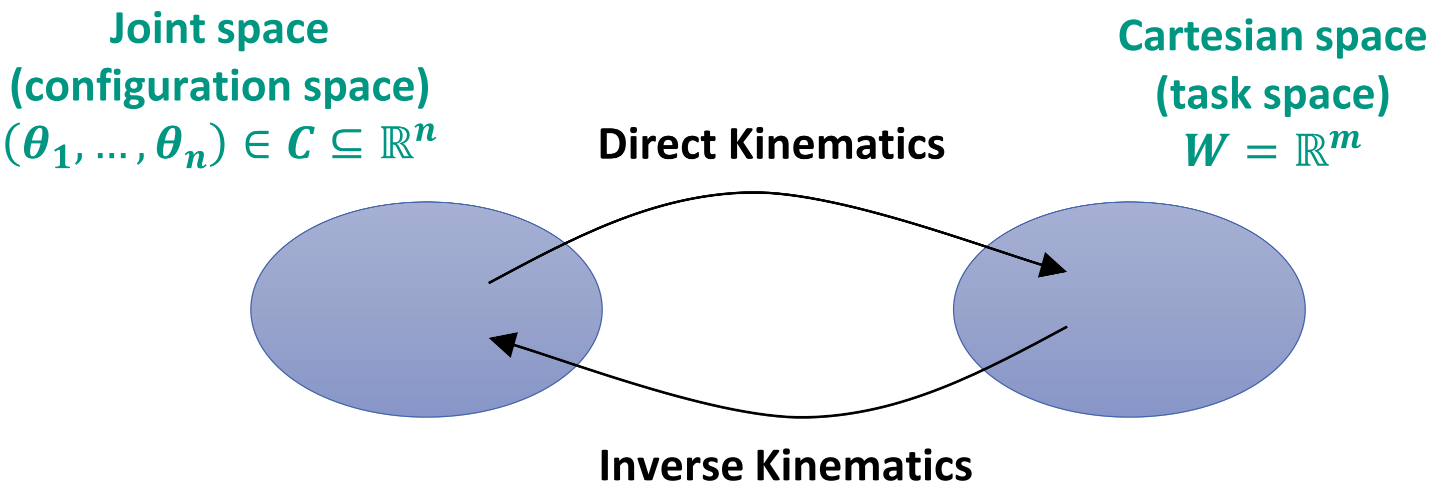
Inverse Kinematics

$\theta = f^{-1}(\boldsymbol{x})$

e.g. position and location of the end effector $\boldsymbol{x}_{TCP} = (x, y, z, \alpha, \beta, \gamma)$

n: Degrees of freedom of movement
m: Degrees of freedom

Inverse Kinematics: Problem Definition



n: Degrees of freedom of movement
m: Degrees of freedom

Inverse Kinematics: Bijection

Direct Kinematics: $\boldsymbol{x} = f(\boldsymbol{\theta}), \quad \boldsymbol{x} \in W, \boldsymbol{\theta} \in C$

Inverse Kinematics: $\boldsymbol{\theta} = f^{-1}(\boldsymbol{x})$

Inverse function f^{-1} only exists if f is **bijective** (injective und surjective)

- Function $f: C \rightarrow W$ is **injective** if for each element in W there is at most one element from C (none at all, exactly one, but not more than one)

$$f(\boldsymbol{\theta}_1) = f(\boldsymbol{\theta}_2) \Rightarrow \boldsymbol{\theta}_1 = \boldsymbol{\theta}_2$$

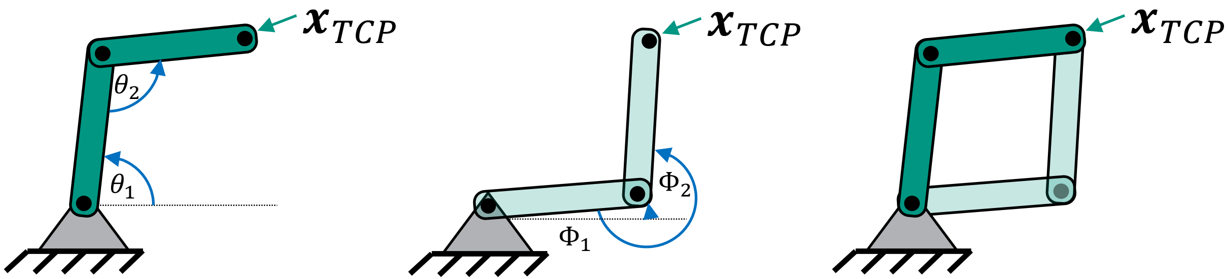
- Function $f: C \rightarrow W$ is **surjective** if for each element in W at least one element from C exists

$$\forall \boldsymbol{x} \in W: \exists \boldsymbol{\theta} \in C: \quad f(\boldsymbol{\theta}) = \boldsymbol{x}$$

In general, the forward kinematics f **is not bijective**

Inverse Kinematics: Injection

Forward kinematics is generally not injective ($f(\boldsymbol{x}_1) = f(\boldsymbol{x}_2) \Rightarrow \boldsymbol{x}_1 = \boldsymbol{x}_2$)



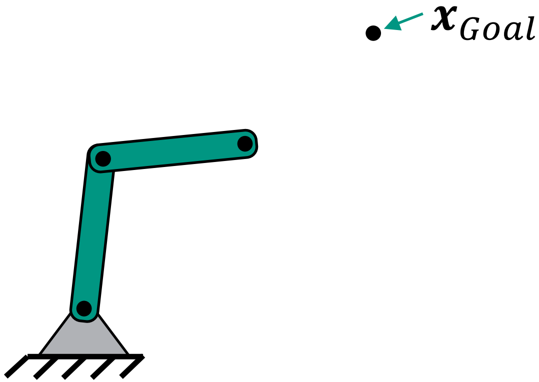
$$\boldsymbol{\theta} = (\theta_1, \theta_2) \in C, \qquad \boldsymbol{\Phi} = (\Phi_1, \Phi_2) \in C$$

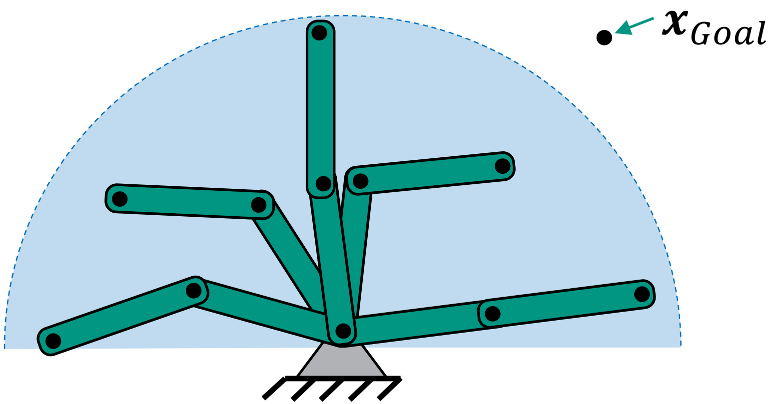
$$f(\boldsymbol{\theta}) = f(\boldsymbol{\Phi}) = \boldsymbol{x}_{TCP}$$

$$f^{-1}(\boldsymbol{x}_{TCP}) = ?, \qquad \boldsymbol{\theta} \text{ or } \boldsymbol{\Phi}?$$

Inverse Kinematics: Surjection

Forward kinematics is generally not surjective ($\forall \boldsymbol{x} \in W\colon \exists \boldsymbol{\theta} \in C\colon \quad f(\boldsymbol{\theta}) = \boldsymbol{x}$)





$$f(\boldsymbol{\theta}) \in \mathbb{R}^2 \qquad f^{-1}(\boldsymbol{x}_{Goal}) = ?$$

There is no $\boldsymbol{\theta} \in C$ for which $f(\boldsymbol{\theta}) = \boldsymbol{x}_{Goal}$.

Can be partially remedied by defining the workspace $W \subset \mathbb{R}^2$.

Inverse Kinematics: Example of a 2 DoF Robot (1)

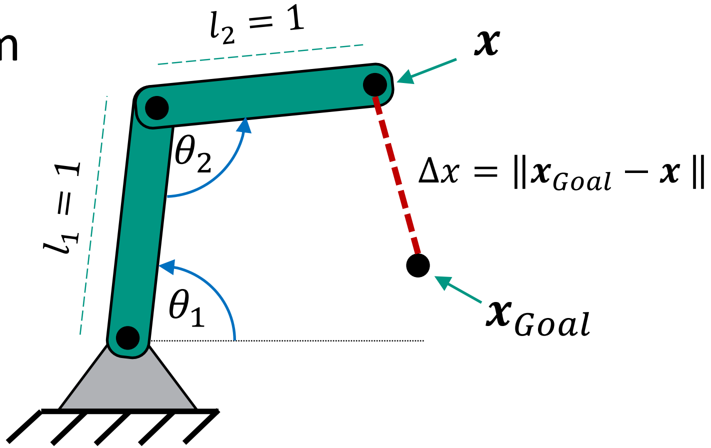
- Position of the end effector (forward kinematics)

$$\boldsymbol{x} = f(\boldsymbol{\theta}) = \begin{pmatrix} \cos\theta_1 + \cos(\theta_1 + \theta_2) \\ \sin\theta_2 + \sin(\theta_1 + \theta_2) \end{pmatrix}$$

- For a given target position $\boldsymbol{x}_{Goal}$ the distance from the current position to the target is:

$$\Delta x = \|\boldsymbol{x}_{Goal} - \boldsymbol{x}\,\|$$

- Inverse kinematics: Find $\boldsymbol{\theta}$ for which $\Delta x = 0$.



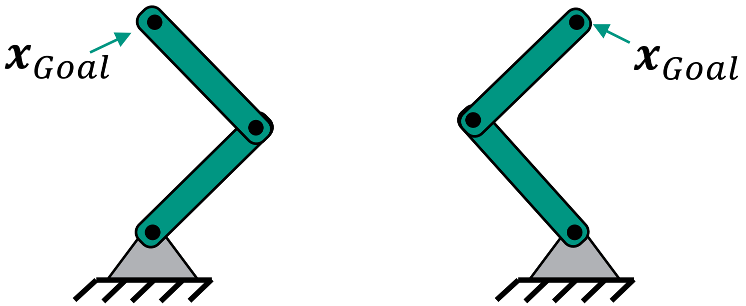
Inverse Kinematics: Example of a 2 DoF Robot (2)

How does the distance change Δx

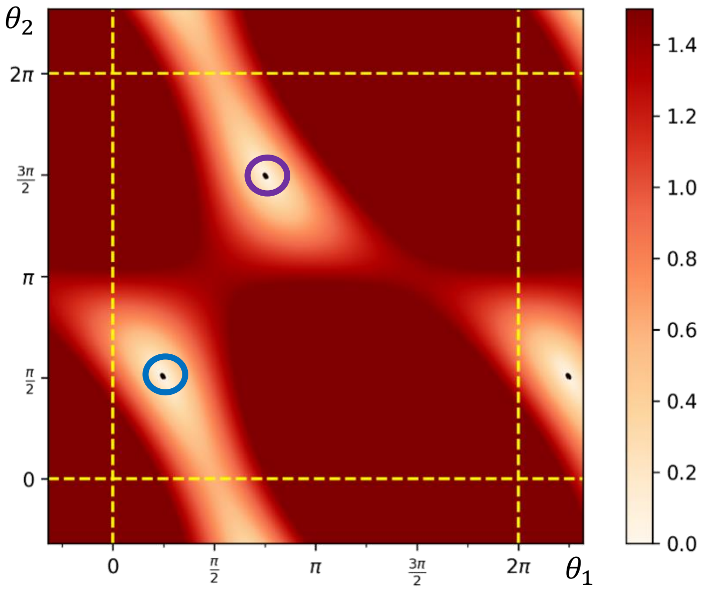
- for different joint angles $\boldsymbol{\theta} = (\theta_1, \theta_2)$?
- for different target positions $\boldsymbol{x}_{Goal}$?

Concrete: $\boldsymbol{x}_{Goal} = \left(0, \sqrt{2}\,\right)^T$

Solutions: $\boldsymbol{\theta_1} = \left(\frac{\pi}{4}, \frac{\pi}{2}\right)$ or $\boldsymbol{\theta_2} = \left(\frac{3\pi}{4}, \frac{3\pi}{2}\right)$

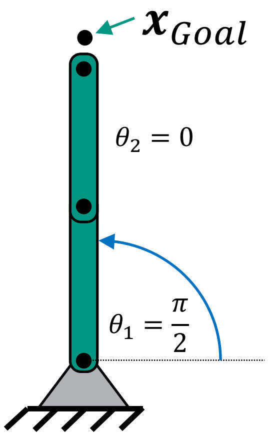


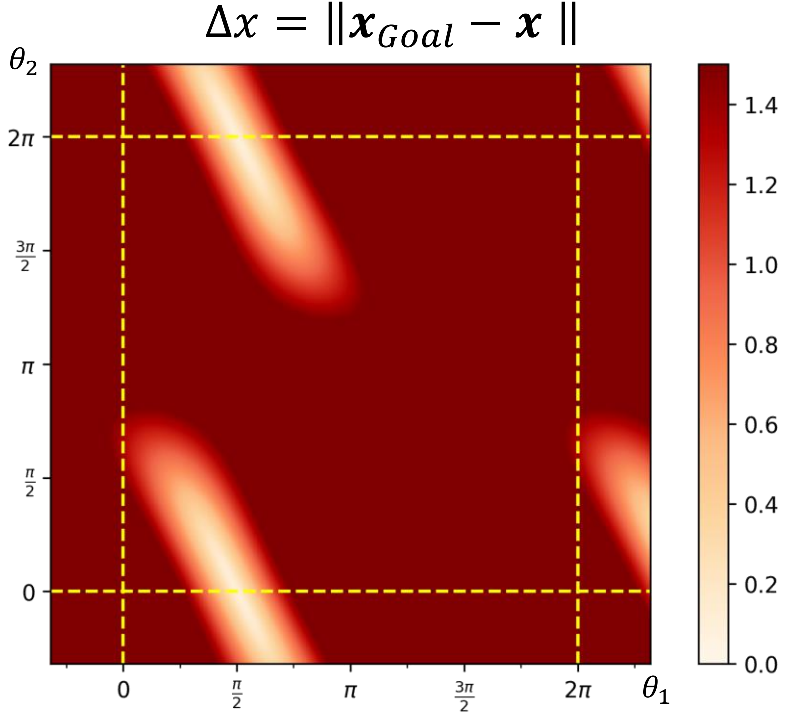
$\Delta x = \|\boldsymbol{x}_{Goal} - \boldsymbol{x}\,\|$



Inverse Kinematics: Example of a 2 DoF Robot (3)

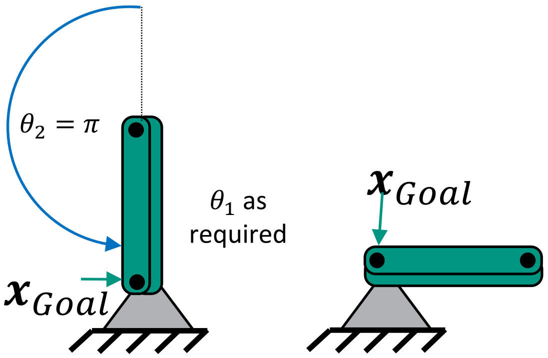
- What happens at $\boldsymbol{x}_{Goal} = (0, 2.1)^T$ outside the workspace?
- No solutions



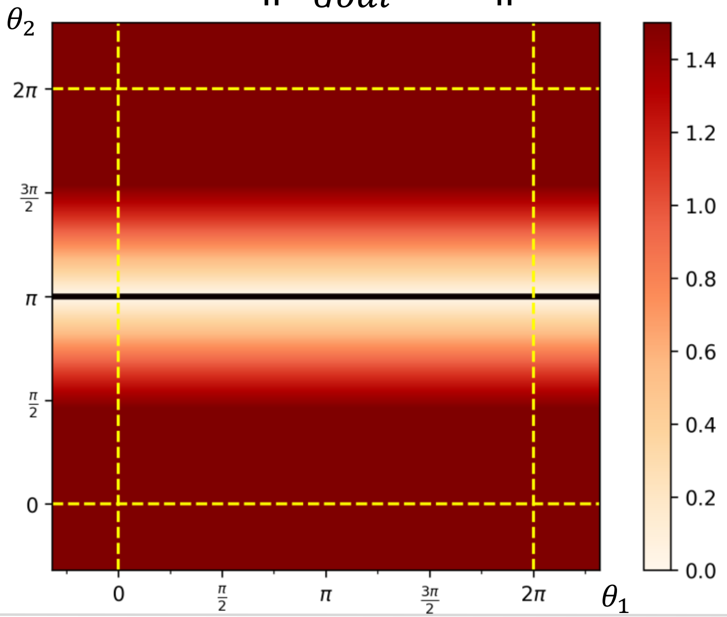


Inverse Kinematics: Example of a 2 DoF Robot (4)

- How does the distance Δx change for $\boldsymbol{x}_{Goal} = (0,0)^T$?
- Infinite number of solutions: $\boldsymbol{\theta} = (\theta_1, \pi)$
  - $\theta_2 = \pi$: The second arm element is folded onto the first arm element
  - θ_1 can be selected as required



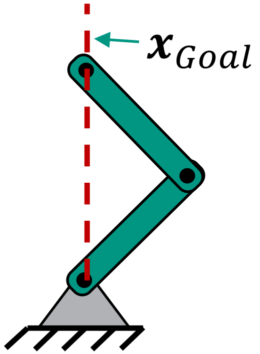
$\Delta x = \|\boldsymbol{x}_{Goal} - \boldsymbol{x}\|$

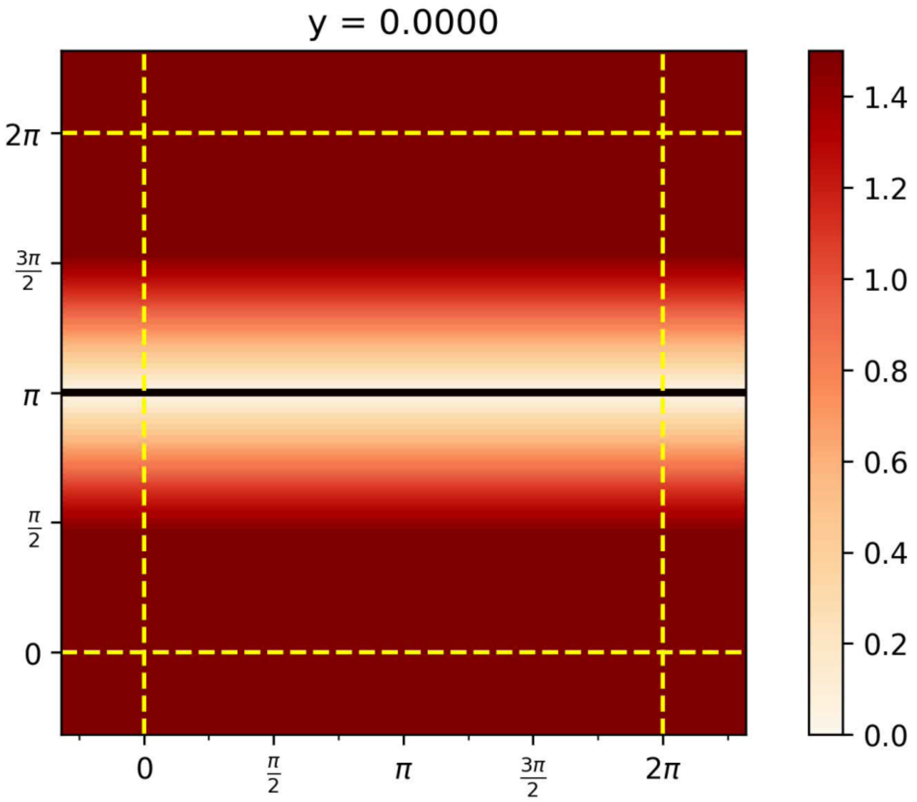


Inverse Kinematics: Example of a 2 DoF Robot (5)

$$\boldsymbol{x}_{Goal} = (0, y)^T$$

$$\Delta x = \|\boldsymbol{x}_{Goal} - \boldsymbol{x}\|$$

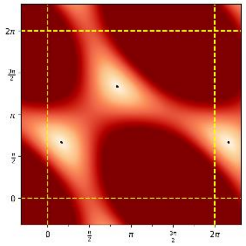




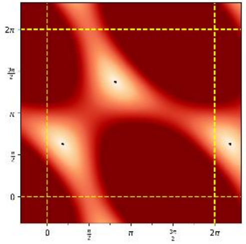
Inverse Kinematics: Example of a 2 DoF Robot (6)

How does Δx change for different target positions $\boldsymbol{x}_{Goal} = \left(0, y_{goal}\right)^T$

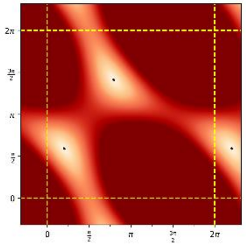
$y_{goal} = 1.0$



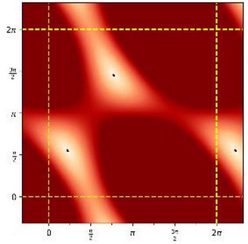
$y_{goal} = 1.1$



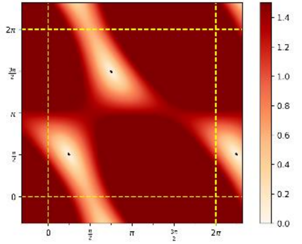
$y_{goal} = 1.2$



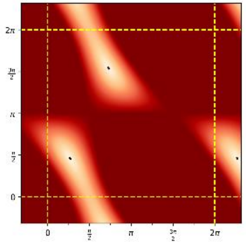
$y_{goal} = 1.3$



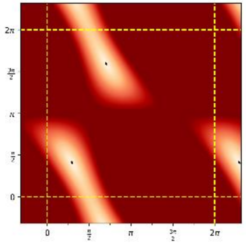
$y_{goal} = 1.4$



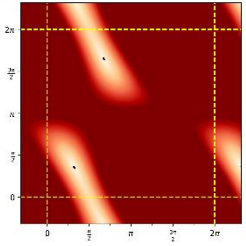
$y_{goal} = 1.5$



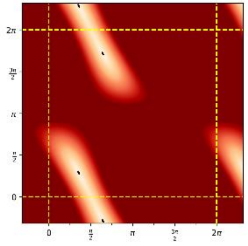
$y_{goal} = 1.6$



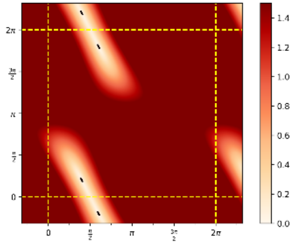
$y_{goal} = 1.7$



$y_{goal} = 1.8$



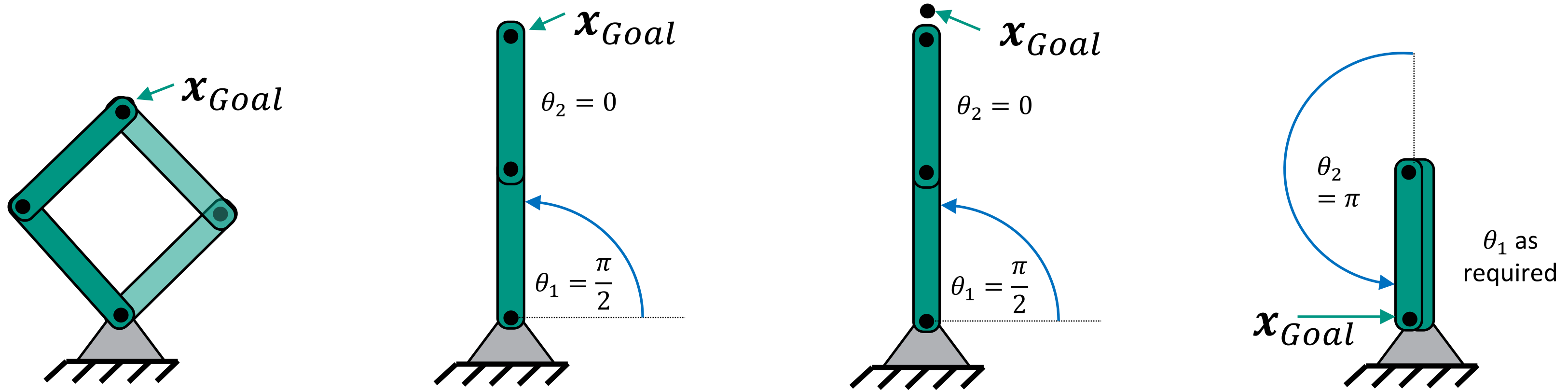
$y_{goal} = 1.9$



Inverse Kinematics: Example of a 2 DoF Robot (7)

In the case of a 2 DoF planar robot, there are four different cases:

- There are **two independent solutions** (normal case).
- There is **exactly one solution** (boundary of the workspace).
- There is **no solution** (outside the workspace).
- There are **infinitely many solutions** (target point in the base).



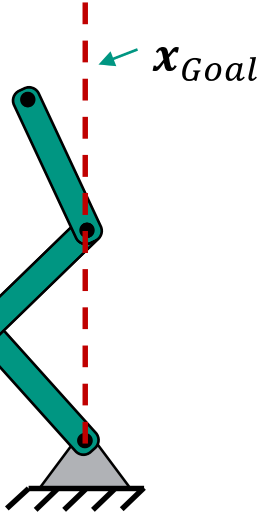
Two solutions | One solution | No solution | Infinitely many solutions

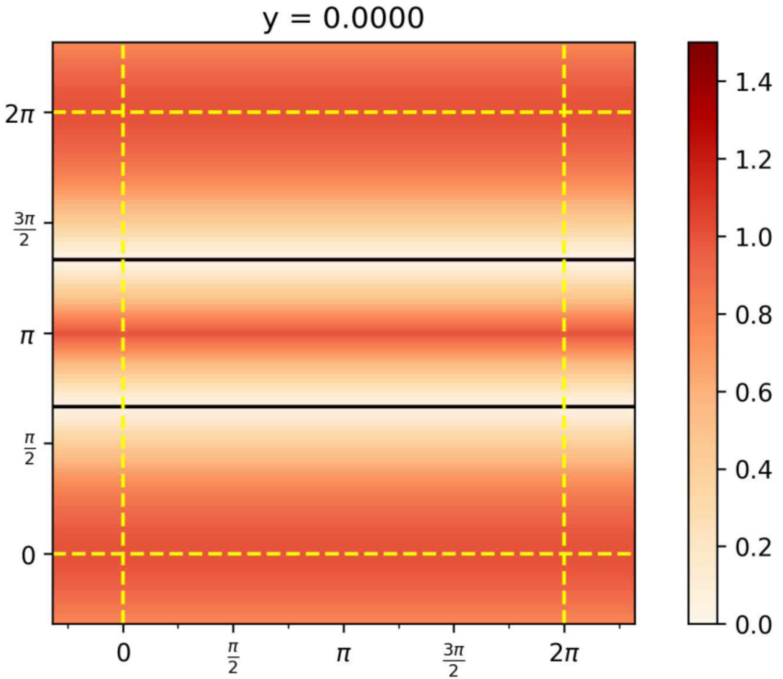
Inverse Kinematics: Example of a 3 DoF Robot

3 DoF robot: What does the solution space look like?

$$\boldsymbol{x}_{Goal} = (0, y)^T$$

$$\Delta x = \|\boldsymbol{x}_{Goal} - \boldsymbol{x}\|$$

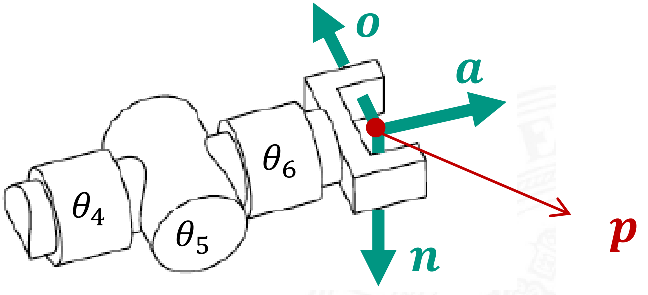




Inverse Kinematics: Procedure

Pose of the TCP

$$T_{TCP} = \begin{pmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Kinematic model:

$$T_{TCP} = {}^{Ref}T_{TCP}(\boldsymbol{\theta}) = A_{0,1}(\theta_1) \cdot A_{1,2}(\theta_2) \cdot \ldots \cdot A_{n-2,n-1}(\theta_{n-1}) \cdot A_{n-1,n}(\theta_n)$$

Given: T_{TCP}, Wanted: $\boldsymbol{\theta}$

Approach: Solve the equation for $\boldsymbol{\theta}$ (**non-linear problem**)

Overview

- Inverse kinematic problem
- Closed-form methods
  - **Geometric**
  - Algebraic
- Numerical methods
  - Gradient descent
  - Jacobian based and pseudoinverse based methods
- Summary

Geometric Method: Procedure

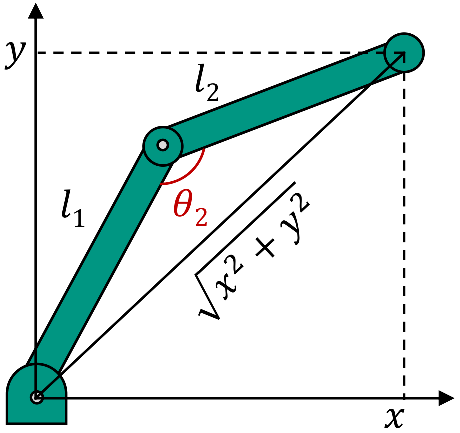
- Use **geometric relationships** to determine the joint angles $\boldsymbol{\theta}$ from the T_{TCP}

- The kinematic model is not used directly.

Application of:

- Trigonometric functions
- Sine / cosine theorems

Geometric Method: Example

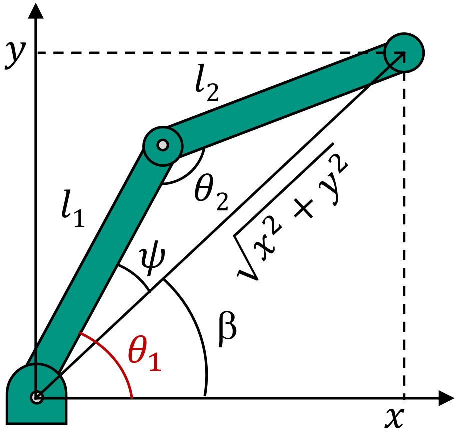


With cosine theorem:

$$x^2 + y^2 = l_1^2 + l_2^2 - 2l_1 l_2 \cos(\theta_2)$$

$$\cos(\theta_2) = -\underbrace{\frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1 l_2}}_{u} \Rightarrow \theta_2 = acos\,(u)$$

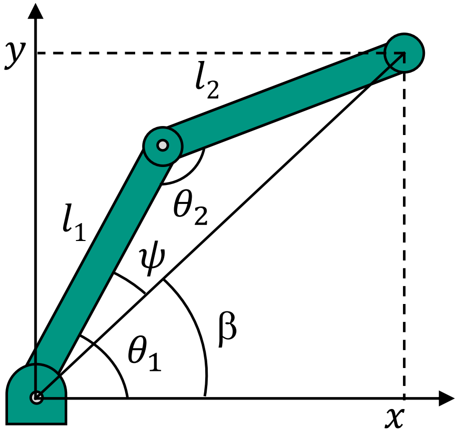
Geometric Method: Example (2)



$$l_2^2 = x^2 + y^2 + l_1^2 - 2l_1\sqrt{x^2+y^2}\cos(\psi)$$

$$\rightarrow \cos(\psi) = \underbrace{\frac{x^2+y^2+l_1^2-l_2^2}{2l_1\sqrt{x^2+y^2}}}_{w} \quad \Rightarrow \quad \psi = acos\,(w)$$

Geometric Method: Example (3)



$$\tan(\beta) = \frac{y}{x} \quad \rightarrow \quad \beta = \operatorname{atan}\left(\frac{y}{x}\right)$$

$$\theta_1 = \psi + \beta$$

Geometric Method: Polynomialization

Transcendental equations are usually difficult to solve, as the variable θ usually appears in the form $\cos\theta$ or $\sin\theta$.

Tool: **Substitution (Tangent half-angle substitution)**

$$u = \tan\left(\frac{\theta}{2}\right)$$

Using:

$$\cos\theta = \frac{1-u^2}{1+u^2} \qquad \sin\theta = \frac{2u}{1+u^2}$$

➔ **Solving polynomial equations**

Overview

- Inverse kinematic problem
- Closed-form methods
  - Geometric
  - **Algebraic**
- Numerical methods
  - Gradient descent
  - Jacobian based and pseudoinverse based methods
- Summary

Algebraic Methods

- Equating the TCP pose T_{TCP} and transformation ${}^{Ref}T_{TCP}$ from the kinematic model:

$$T_{TCP} = {}^{Ref}T_{TCP}(\boldsymbol{\theta})$$

- **Comparison of the coefficients** of the two matrices

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nn} \end{pmatrix} \Rightarrow a_{ij} = b_{ij} \qquad \forall i,j \in [1{:}n]$$

- 16 equations for homogeneous matrices in 3D (4 trivial: $0 = 0$, $1 = 1$)

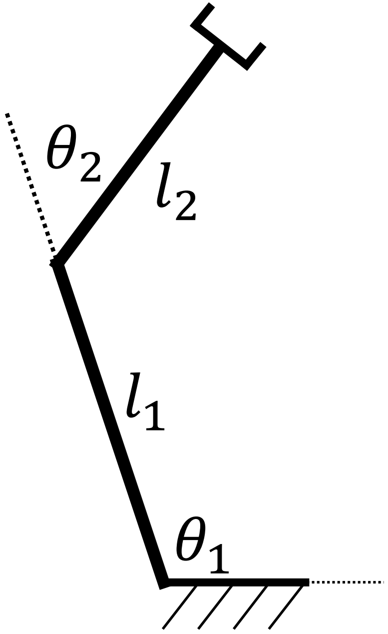
  ➔ **12 non-trivial equations**

Algebraic Methods: Example (1)

- From **kinematic model**

$$^{Ref}T_{TCP} = \begin{pmatrix} c_{12} & -s_{12} & 0 & l_1 c_1 + l_2 c_{12} \\ s_{12} & c_{12} & 0 & l_1 s_1 + l_2 s_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$c_{12} = \cos(\theta_1 + \theta_2);\ \ s_{12} = sin(\theta_1 + \theta_2)$$



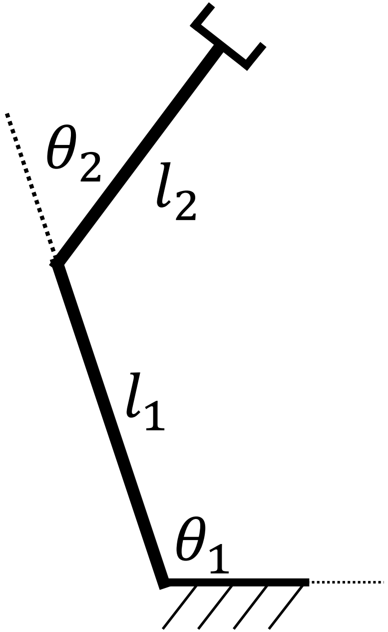
- Desired position of the end effector in space:
  Position (x, y), orientation(ϕ)

$$P_{TCP} = \begin{pmatrix} c_\phi & -s_\phi & 0 & x \\ s_\phi & c_\phi & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Algebraic Methods: Example (2)

Coefficient comparison

$$\begin{pmatrix} c_\phi & -s_\phi & 0 & x \\ s_\phi & c_\phi & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} c_{12} & -s_{12} & 0 & l_1 c_1 + l_2 c_{12} \\ s_{12} & c_{12} & 0 & l_1 s_1 + l_2 s_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



$$c_\phi = c_{12} \quad (1)$$
$$s_\phi = s_{12} \quad (2)$$
$$x = l_1 c_1 + l_2 c_{12} \quad (3)$$
$$y = l_1 s_1 + l_2 s_{12} \quad (4)$$

➔ Resolve for $\boldsymbol{\theta}$

Algebraic Methods: Example (3)

Sum of the squares of (3) and (4)

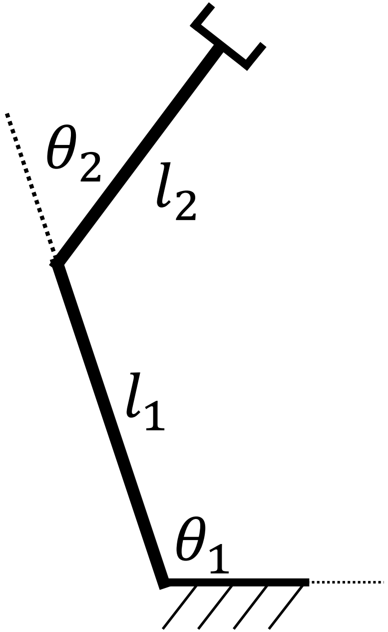
$$x^2 = l_1^2 c_1^2 + 2 l_1 c_1 l_2 c_{12} + l_2^2 c_{12}^2$$
$$y^2 = l_1^2 s_1^2 + 2 l_1 s_1 l_2 s_{12} + l_2^2 s_{12}^2$$

$$s_1^2 + c_1^2 = 1; \quad s_{12}^2 + c_{12}^2 = 1$$

$$x^2 + y^2 = l_1^2 + l_2^2 + 2 l_1 l_2 (c_1 c_{12} + s_1 s_{12}) = l_1^2 + l_2^2 + 2 l_1 l_2 c_2$$

$$c_2 = \frac{x^2 + y^2 - l_1^2 - l_2^2}{2 l_1 l_2} \rightarrow \theta_2$$

Two solutions for θ_2 are possible. Why?
➔ **Redundancy**



Algebraic Methods: Example (4)

Calculation of θ_1

- Coefficient comparison:
$$x = l_1 c_1 + l_2 c_{12}, \qquad y = l_1 s_1 + l_2 s_{12}$$
- Addition theorem: $\cos(\theta_1 + \theta_2) = \cos(\theta_1)\cos(\theta_2) - \sin(\theta_1)\sin(\theta_2)$
$$x = l_1 c_1 + l_2(c_1 c_2 - s_1 s_2)$$
$$y = l_1 s_1 + l_2(s_1 c_2 + s_2 c_1)$$
- Simplify:
$$x = (l_1 + l_2 c_2)c_1 - (l_2 s_2)s_1$$
$$y = (l_1 + l_2 c_2)s_1 + (l_2 s_2)c_1$$

Resolution difficult.
Help with templates for typical equations or symbolic math in Matlab, Maple, Mathematica.

Algebraic Methods: Solution algorithm

Problem:
Often not all joint angles can be determined from the 12 equations.

Approach:
Knowledge of the transformations increases the chance of solving the equations.

Given:
The transformation matrices $A_{0,1} \cdot A_{1,2} \cdot \ldots \cdot A_{n-1,n}$ and T_{TCP}

Wanted:
The joint angles θ_1 to θ_n

Algebraic Methods: Procedure

$$T_{TCP} = A_{0,1}(\theta_1) \cdot A_{1,2}(\theta_2) \cdot A_{2,3}(\theta_3) \cdot A_{3,4}(\theta_4) \cdot A_{4,5}(\theta_5) \cdot A_{5,6}(\theta_6)$$

Algebraic Methods: Procedure

Karlsruhe Institute of Technology

- Starting point: the matrix equation

$$T_{TCP} = A_{0,1}(\theta_1) \cdot A_{1,2}(\theta_2) \cdot A_{2,3}(\theta_3) \cdot A_{3,4}(\theta_4) \cdot A_{4,5}(\theta_5) \cdot A_{5,6}(\theta_6)$$

- Procedure:
  1. Invert $A_{0,1}(\theta_1)$ and multiply both sides of the equation by $A_{0,1}^{-1}$
  2. Try to find an equation from the newly created system of equations that contains only one unknown and solve this equation for the unknown.
  3. Try to find an equation in the system of equations that can be solved by substituting the solution found in the last step for one unknown.
  4. If no more solutions can be found, another matrix $(A_{1,2}(\theta_2))$ must be inverted.
  5. Repeat steps 1 - 4 until all joint angles have been determined.

Algebraic Methods: Equations

$$\boldsymbol{T_{TCP} = A_{0,1} \cdot A_{1,2} \cdot A_{2,3} \cdot A_{3,4} \cdot A_{4,5} \cdot A_{5,6}}$$

$$A_{0,1}^{-1} \cdot T_{TCP} = A_{1,2} \cdot A_{2,3} \cdot A_{3,4} \cdot A_{4,5} \cdot A_{5,6}$$

$$A_{1,2}^{-1} \cdot A_{0,1}^{-1} \cdot T_{TCP} = A_{2,3} \cdot A_{3,4} \cdot A_{4,5} \cdot A_{5,6}$$

$$A_{2,3}^{-1} \cdot A_{1,2}^{-1} \cdot A_{0,1}^{-1} \cdot T_{TCP} = A_{3,4} \cdot A_{4,5} \cdot A_{5,6}$$

$$A_{3,4}^{-1} \cdot A_{2,3}^{-1} \cdot A_{1,2}^{-1} \cdot A_{0,1}^{-1} \cdot T_{TCP} = A_{4,5} \cdot A_{5,6}$$

$$A_{4,5}^{-1} \cdot A_{3,4}^{-1} \cdot A_{2,3}^{-1} \cdot A_{1,2}^{-1} \cdot A_{0,1}^{-1} \cdot T_{TCP} = A_{5,6}$$

12 non-trivial equations from each matrix equation

$$T_{TCP} \cdot A_{5,6}^{-1} = A_{0,1} \cdot A_{1,2} \cdot A_{2,3} \cdot A_{3,4} \cdot A_{4,5}$$

$$T_{TCP} \cdot A_{5,6}^{-1} \cdot A_{4,5}^{-1} = A_{0,1} \cdot A_{1,2} \cdot A_{2,3} \cdot A_{3,4}$$

$$T_{TCP} \cdot A_{5,6}^{-1} \cdot A_{4,5}^{-1} \cdot A_{3,4}^{-1} = A_{0,1} \cdot A_{1,2} \cdot A_{2,3}$$

$$T_{TCP} \cdot A_{5,6}^{-1} \cdot A_{4,5}^{-1} \cdot A_{3,4}^{-1} \cdot A_{2,3}^{-1} = A_{0,1} \cdot A_{1,2}$$

$$T_{TCP} \cdot A_{5,6}^{-1} \cdot A_{4,5}^{-1} \cdot A_{3,4}^{-1} \cdot A_{2,3}^{-1} \cdot A_{1,2}^{-1} = A_{0,1}$$

Overview

- Inverse kinematic problem
- Closed-form methods
  - Geometric
  - **Algebraic**
- Numerical methods
  - Gradient descent
  - Jacobian based and pseudoinverse based methods
- Summary

Numerical Methods: Jacobian Matrix (Repetition)

Given a differentiable function $f\colon \mathbb{R}^n \to \mathbb{R}^m$

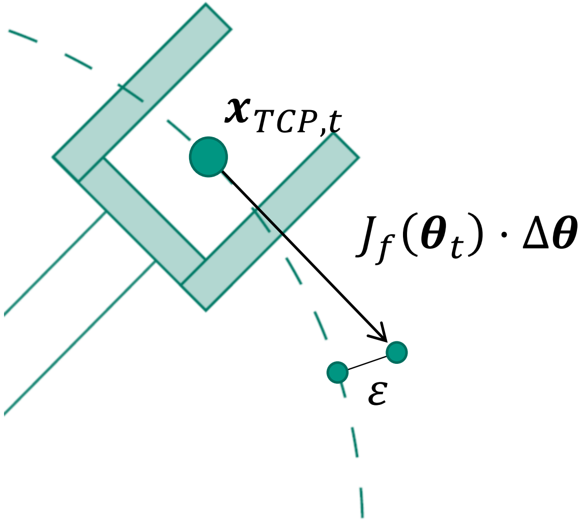
The **Jacobian matrix** contains all first-order partial derivatives of f. For an $\boldsymbol{a} \in \mathbb{R}^n$ the following applies:

$$J_f(\boldsymbol{a}) = \left(\frac{\partial f_i}{\partial x_j}(\boldsymbol{a})\right)_{i,j} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\boldsymbol{a}) & \cdots & \frac{\partial f_1}{\partial x_n}(\boldsymbol{a}) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(\boldsymbol{a}) & \cdots & \frac{\partial f_m}{\partial x_n}(\boldsymbol{a}) \end{pmatrix} \in \mathbb{R}^{m \times n}$$

The following applies:

$$\boxed{\dot{\boldsymbol{x}}(t) = \frac{df(\theta(t))}{dt} = J_f\big(\theta(t)\big) \cdot \dot{\theta}(t)}$$

Numerical Methods



TCP pose via **forward kinematics**:

$$\boldsymbol{x}_{TCP,t} = f(\boldsymbol{\theta}_t)$$

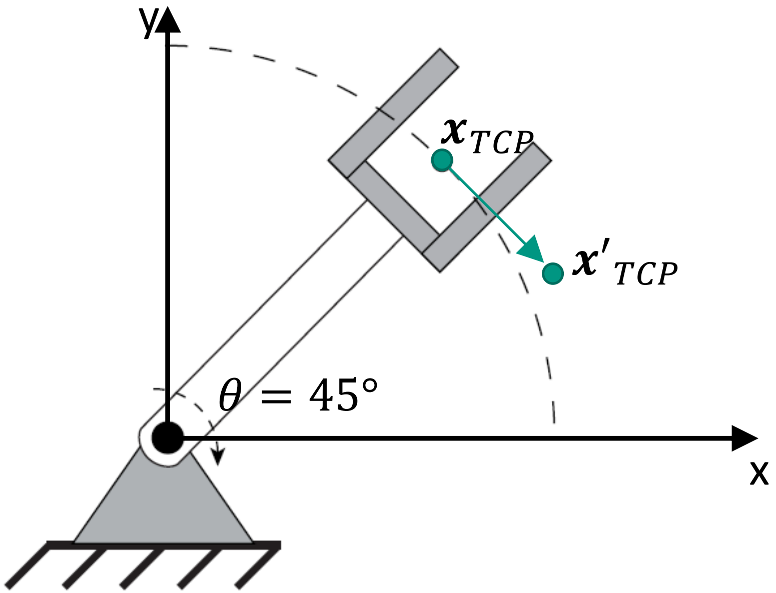
Jacobian matrix provides **movement tangents** in the current position $\boldsymbol{\theta}_t$:

$$J_f(\boldsymbol{\theta}_t) = \frac{\partial f(\boldsymbol{\theta}_t)}{\partial \boldsymbol{\theta}_t}$$

Assumption: Model valid for small $\Delta\theta$

- **Linear approximation** of the movement
- Approximation error ε exists

Numerical Methods: Example



$$J_f(\theta) = J_f(45°) = s \cdot \begin{pmatrix} 1 \\ -1 \end{pmatrix}, s \in \mathbb{R}$$

Overview

Inverse kinematic problem

Closed-form methods

- Geometric
- Algebraic

Numerical methods

- **Gradient descent**
- Jacobian based and pseudoinverse based methods

Summary

Gradient Descent: Optimization Problem

- Forwards kinematics:

$$\boldsymbol{x} = f(\boldsymbol{\theta}), \qquad \boldsymbol{x} \in W \subset \mathbb{R}^m, \qquad \boldsymbol{\theta} \in C \subset \mathbb{R}^n$$

- Error function for target pose $\boldsymbol{x}_{Goal} \in W$:

$$\mathrm{e}(\boldsymbol{\theta}) = \|\boldsymbol{x}_{Goal} - f(\boldsymbol{\theta})\|^2$$

- Solutions for inverse kinematics for: $e(\boldsymbol{\theta}) = 0$

- Approach: **Gradient descent**

Gradient Descent: Derivation of the Error Function

Error function for target pose $\boldsymbol{x}_{Goal} \in W$: $\quad \mathrm{e}(\boldsymbol{\theta}) = \|\boldsymbol{x}_{Goal} - f(\boldsymbol{\theta})\|^2$

$$F(\boldsymbol{x}) = \|A\boldsymbol{x} - \boldsymbol{b}\|^2$$
$$\nabla F(\boldsymbol{x}) = 2\,A^T(A\boldsymbol{x} - \boldsymbol{b})$$

Derivation with chain rule:

$$\frac{\partial \mathrm{e}}{\partial \boldsymbol{\theta}} = \frac{\partial(\|\boldsymbol{x}_{Goal} - f(\boldsymbol{\theta})\|^2)}{\partial\left(\boldsymbol{x}_{Goal} - f(\boldsymbol{\theta})\right)} \cdot \frac{\partial\left(\boldsymbol{x}_{Goal} - f(\boldsymbol{\theta})\right)}{\partial \boldsymbol{\theta}}$$

Note: $\frac{\partial \mathrm{e}}{\partial \boldsymbol{\theta}} \in \mathbb{R}^{1\times m}$ is a row vector

$$\frac{\partial \mathrm{e}}{\partial \boldsymbol{\theta}} = -2 \cdot \left(\boldsymbol{x}_{Goal} - f(\boldsymbol{\theta})\right)^T \cdot J(\boldsymbol{\theta})$$

$$(\boldsymbol{x}^T \cdot A)^T = A^T \cdot \boldsymbol{x}$$

$$\left(\frac{\partial \mathrm{e}}{\partial \boldsymbol{\theta}}\right)^T = 2 \cdot J^T(\boldsymbol{\theta}) \cdot (f(\boldsymbol{\theta}) - \boldsymbol{x}_{Goal})$$

Gradient Descent: Algorithm

Error function for target pose $\boldsymbol{x}_{Goal} \in W$: $\quad \mathrm{e}(\boldsymbol{\theta}) = \| \boldsymbol{x}_{Goal} - f(\boldsymbol{\theta}) \|^2$

Gradient: $grad(e) = \frac{\partial \mathrm{e}}{\partial \boldsymbol{\theta}} = 2 \cdot J^T(\boldsymbol{\theta}) \cdot (f(\boldsymbol{\theta}) - \boldsymbol{x}_{Goal})$

Select start configuration: $\boldsymbol{\theta}_0 \in C, i = 0$

Step length $\gamma \in \mathbb{R}$

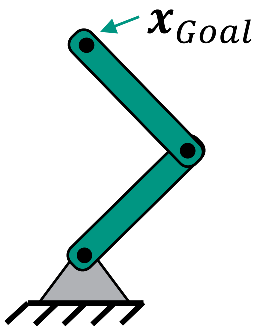
As long as $\mathrm{e}(\boldsymbol{\theta}_i) > e_{Threshold}$: # Limit value

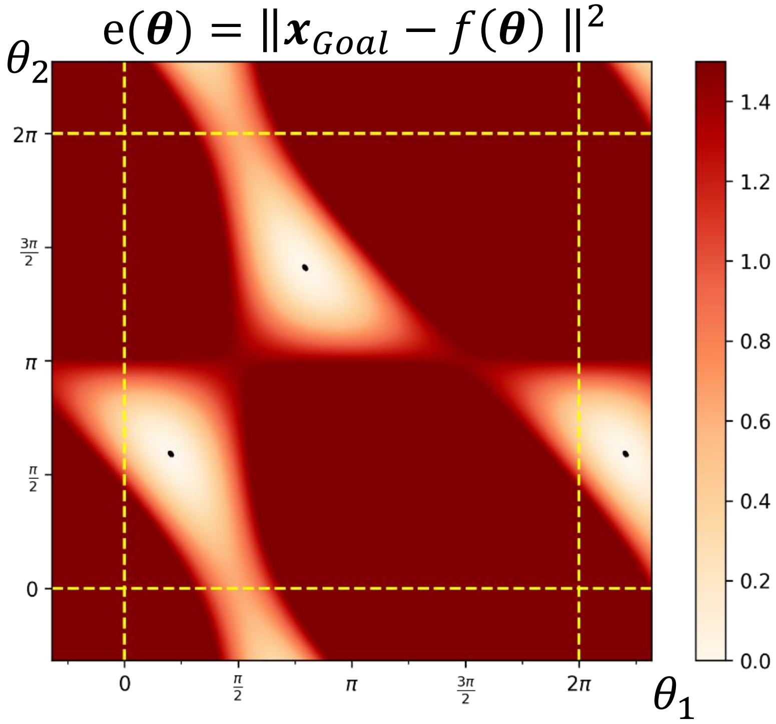
$\quad \boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \gamma \cdot 2 \cdot J^T(\boldsymbol{\theta}_i) \cdot (f(\boldsymbol{\theta}_i) - \boldsymbol{x}_{Goal})$ # − Gradient

$\quad i = i + 1$

Gradient Descent: Example (1)

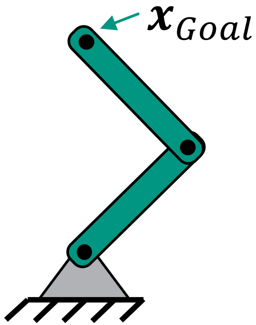
- 2-DoF planar robot: $\boldsymbol{\theta} = (\theta_1, \theta_2) \in C$
- Target pose $\boldsymbol{x}_{Goal} = (0, 1.2)^T$

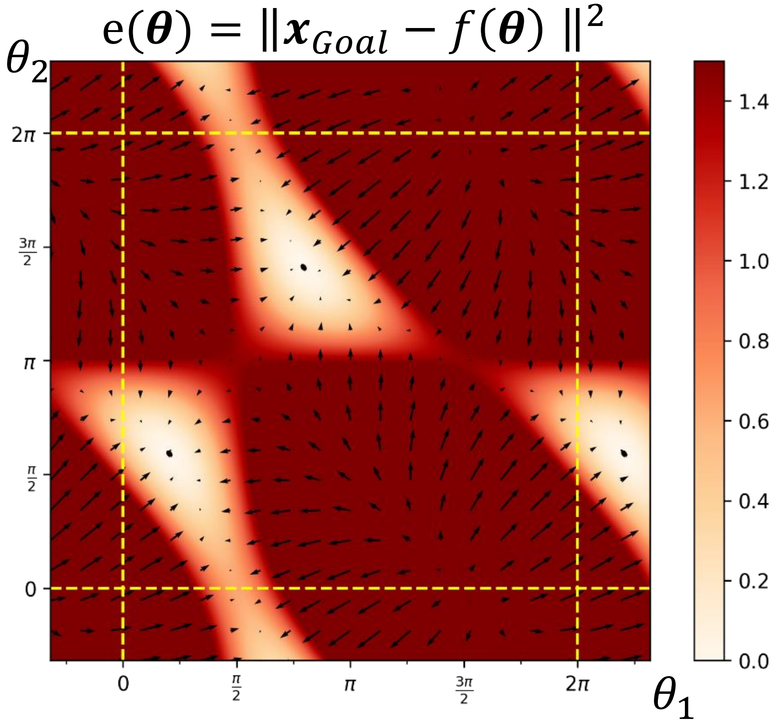




Gradient Descent: Example (2)

- 2-DoF planar robot: $\boldsymbol{\theta} = (\theta_1, \theta_2) \in C$
- Target pose $\boldsymbol{x}_{Goal} = (0, 1.2)^T$
- Gradient field



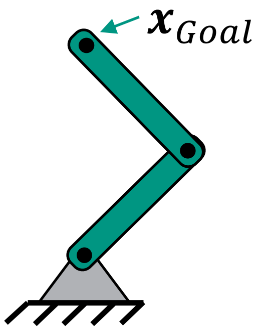


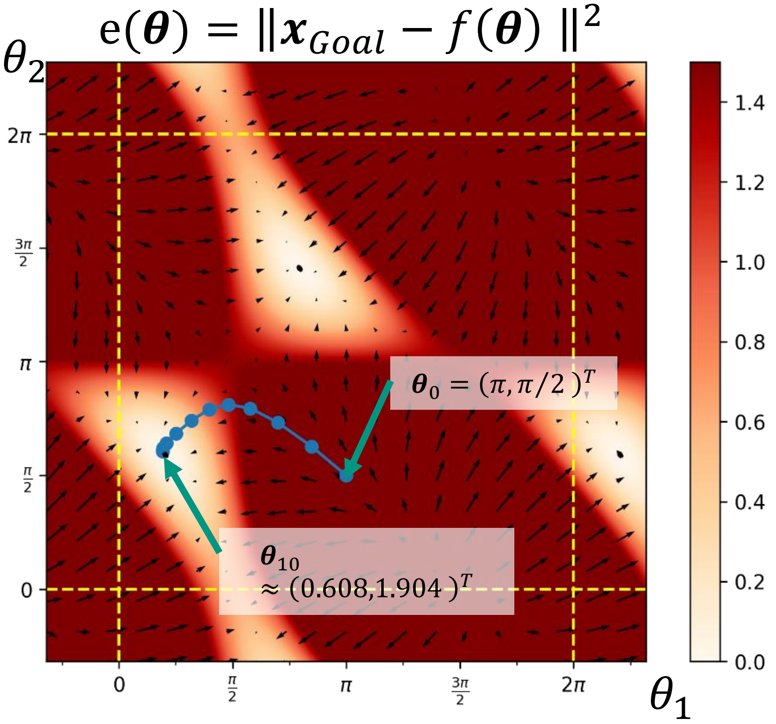
Gradient Descent: Example (3)

2-DoF planar robot: $\boldsymbol{\theta} = (\theta_1, \theta_2) \in C$
Target pose $\boldsymbol{x}_{Goal} = (0, 1.2)^T$
Gradient field

Step length: $\gamma = 0.2$

Start: $\boldsymbol{\theta}_0 = \left(\pi, \frac{\pi}{2}\right)^T$

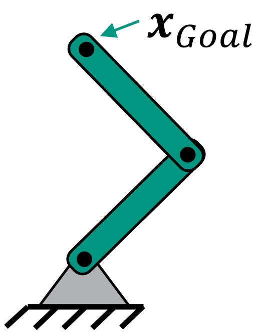


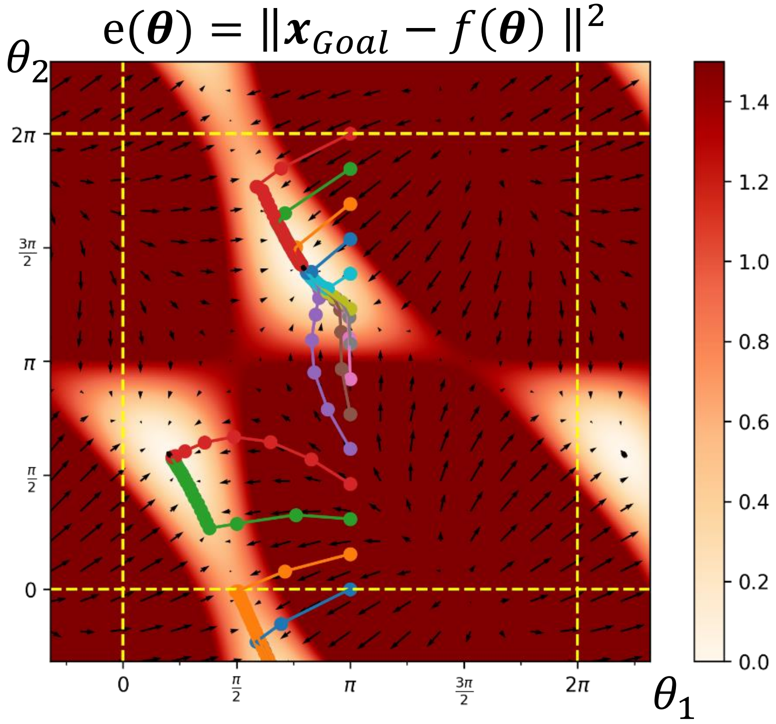


Gradient Descent: Example (4)

2-DoF planar robot: $\boldsymbol{\theta} = (\theta_1, \theta_2) \in C$
Target pose $\boldsymbol{x}_{Goal} = (0, 1.2)^T$
Gradient field

Step length: $\gamma = 0.2$
Different starting points:
$\boldsymbol{\theta}_0 = \left(\pi, \theta_{2,start}\right)^T$





Overview

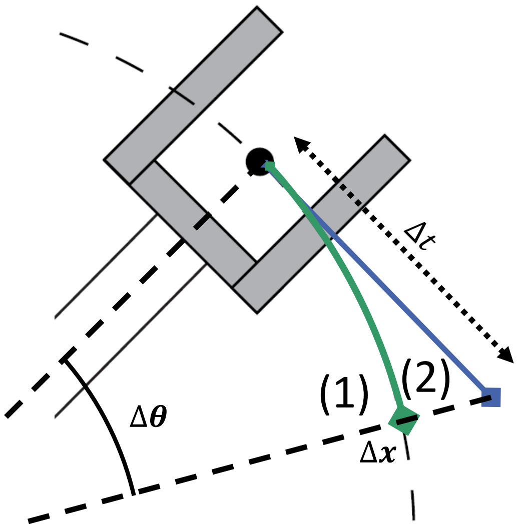
- Inverse kinematic problem

- Closed-form methods
  - Geometric
  - Algebraic

- Numerical methods
  - Gradient descent
  - **Jacobian based and pseudoinverse based methods**

- Summary

Numerical Methods: Difference Quotient



1) Actual movement according to:

$$\dot{\boldsymbol{x}}(t) = J(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}(t)$$

2) Approximate movement in the interval Δt using the difference quotient:

$$\Delta\boldsymbol{x} \approx J(\boldsymbol{\theta})\Delta\boldsymbol{\theta}$$

Approximation of the change by transition from the differential quotient to the **difference quotient**

Linearization of the problem

Numerical Methods: Inversion

- Achieved so far: **Local, linear approach to forward kinematics**

$$\Delta \boldsymbol{x} = \boldsymbol{f}(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}) - \boldsymbol{f}(\boldsymbol{\theta}) \approx J_f(\boldsymbol{\theta}) \cdot \Delta\boldsymbol{\theta}$$

- **Wanted:** Solution for the inverse problem

$$\Delta\boldsymbol{\theta} \approx \boldsymbol{g}(\Delta\boldsymbol{x}) = J_f^{-1}(\boldsymbol{\theta}) \cdot \Delta\boldsymbol{x}$$

- Inversion is possible if:
  - $J_f(\boldsymbol{\theta})$ is quadratic (Non-redundant robots, d.h. $n = m$)
  - $J_f(\boldsymbol{\theta})$ has full rank

Numerical Methods: Pseudoinverse

Pseudoinverse: Generalization of the inverse matrix to singular and non-square matrices $A \in \mathbb{R}^{m \times n}$ **(redundant robots)**

Definition: Moore-Penrose Pseudoinverse (with full line rank*)

$$A^{+} = A^{T}(AA^{T})^{-1}$$

The following apply:

$(A^{+})^{+} = A$

$(A^{T})^{+} = (A^{+})^{T}$

$(\lambda A)^{+} = \lambda^{-1} A^{+}$, for a $\lambda \neq 0$

*Full line rank is usually given for J_f. Exception: **singularities!**

Pseudoinverse: Derivation

Calculate the best possible solution of a system of linear equations in terms of the sum of least squares.

$$A\boldsymbol{x} = \boldsymbol{b} \qquad \text{// } A \text{ is a rectangular matrix, not invertible}$$

$$A^T A\boldsymbol{x} = A^T\boldsymbol{b} \qquad \text{// } A^T A \text{ is a square matrix, invertible}$$

$$\underbrace{(A^T A)^{-1} A^T A}_{I}\, \boldsymbol{x} = (A^T A)^{-1} A^T \boldsymbol{b}$$

$$\widehat{\boldsymbol{x}} = \underbrace{(A^T A)^{-1} A^T}_{A^+}\ \boldsymbol{b} \qquad \text{// } \widehat{\boldsymbol{x}} \text{ is a least squares solution of } A\boldsymbol{x} = \boldsymbol{b}$$

$$\widehat{\boldsymbol{x}} = A^+\boldsymbol{b} \qquad \text{// } A^+ \text{ is the pseudo-inverse of } A$$

Pseudoinverse

- The pseudo inverse minimizes the error $\left\| J_f \dot{\boldsymbol{\theta}} - \dot{\boldsymbol{x}} \right\|^2$; it finds the norm-minimal solution $\left\| \dot{\boldsymbol{\theta}} \right\|^2$

$$\min_{\dot{\boldsymbol{\theta}}} \left\| J_f \dot{\boldsymbol{\theta}} - \dot{\mathbf{x}} \right\|^2 = \min_{\dot{\boldsymbol{\theta}}} \left(J_f \dot{\boldsymbol{\theta}} - \dot{\mathbf{x}} \right)^T \left(J_f \dot{\boldsymbol{\theta}} - \dot{\mathbf{x}} \right)$$

$$\nabla_{\dot{\boldsymbol{\theta}}} \left\| J_f \dot{\boldsymbol{\theta}} - \dot{\mathbf{x}} \right\|^2 = 2 J_f^T \left(J_f \dot{\boldsymbol{\theta}} - \dot{\mathbf{x}} \right) = 0$$

$$\dot{\boldsymbol{\theta}} = \underbrace{\left(J_f^T J_f \right)^{-1} J_f^T}_{J_f^+} \dot{x}$$

$$A^+ = \left(A^T A \right)^{-1} A^T$$

Pseudoinverse: Summary

1. Forward kinematics as a function:
   $$\boldsymbol{x}(t) = f(\boldsymbol{\theta}(t))$$

2. Derivation with respect to time:
   $$\frac{d\boldsymbol{x}(t)}{dt} = \dot{\boldsymbol{x}}(t) = J_f(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}(t)$$

3. Transition to the difference quotient:
   $$\Delta\boldsymbol{x} \approx J_f(\boldsymbol{\theta})\Delta\boldsymbol{\theta}$$

4. Reverse: $\Delta\boldsymbol{\theta} \approx J_f^{+}(\boldsymbol{\theta})\Delta\boldsymbol{x}$

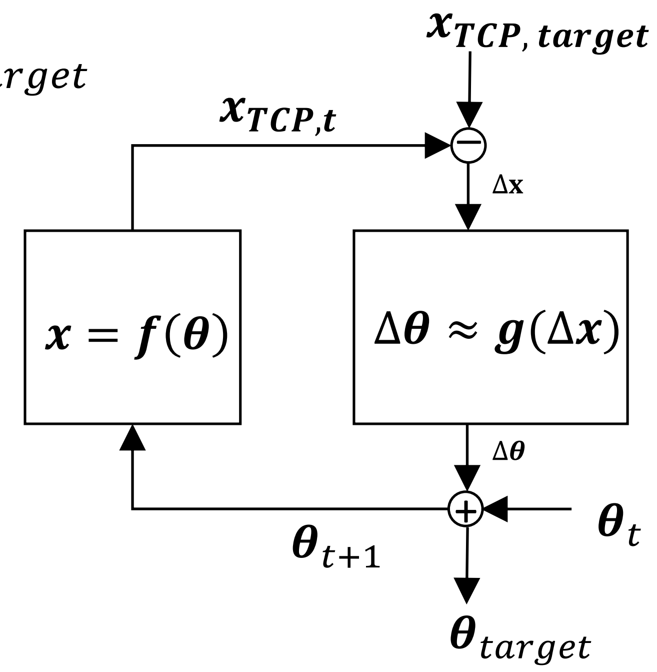
$\boldsymbol{x}(t) \in \mathbb{R}^6$: TCP-pose
$\boldsymbol{\theta}(t) \in \mathbb{R}^n$: Joint angle positions

$\dot{\boldsymbol{x}}(t) \in \mathbb{R}^6$: TCP-velocities
$\dot{\boldsymbol{\theta}}(t) \in \mathbb{R}^n$: Joint velocities
$J_f(\boldsymbol{\theta}) \in \mathbb{R}^{6\times n}$: Jacobian Matrix

$\Delta\boldsymbol{x} \in \mathbb{R}^6$: Errors in TCP-Pose
$\Delta\boldsymbol{\theta} \in \mathbb{R}^n$: Errors in joint positions

Pseudoinverse: Iterative Approach

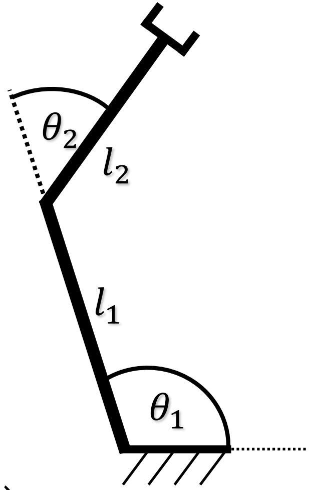
- **Given:** Target pose of the TCP $\boldsymbol{x}_{TCP,\,target}$
- **Wanted:** Joint angle vector $\boldsymbol{\theta}$ that realizes $x_{TCP,\,target}$
- **Iterative approach** starting with initial configuration $\boldsymbol{\theta}_0$ and $\boldsymbol{x}_{TCP,0}$
  1. Calculate $\boldsymbol{x}_{TCP,t}$ in iteration t from joint angle positions $\boldsymbol{\theta}_t$
  2. Calculate error $\Delta\boldsymbol{x}$ from $\boldsymbol{x}_{TCP,\,target}$ and calculated $\boldsymbol{x}_{TCP,t}$
  3. Use approximated inverse kinematic model $\boldsymbol{g}$ to calculate joint angle error $\Delta\boldsymbol{\theta}$
  4. Calculate $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \Delta\boldsymbol{\theta}$
  5. Continue with iteration t+1



Pseudoinverse: Example Calculation (1)

- Position:

$$x = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2)$$
$$y = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2)$$



- Velocity:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = J_f(\boldsymbol{\theta}) \cdot \dot{\boldsymbol{\theta}}(t) = J_f(\boldsymbol{\theta}) \cdot \begin{pmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{pmatrix} =$$

$$= \begin{pmatrix} -l_1 \sin\theta_1 - l_2 \sin(\theta_1 + \theta_2) & -l_2 \sin(\theta_1 + \theta_2) \\ l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) & l_2 \cos(\theta_1 + \theta_2) \end{pmatrix} \cdot \begin{pmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{pmatrix}$$

Pseudoinverse: Example Calculation (2)

- The Jacobian matrix must be inverted:

$$\begin{pmatrix} \Delta\theta_1 \\ \Delta\theta_2 \end{pmatrix} = \underbrace{\frac{1}{l_1 l_2 \sin\theta_2} \begin{pmatrix} l_2 c_{12} & l_2 s_{12} \\ -l_2 c_{12} - l_1 c_1 & -l_1 s_{12} - l_1 s_1 \end{pmatrix}}_{J_f(\boldsymbol{\theta})^{-1}} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$

Abbreviations:

$c_{12} = \cos(\theta_1 + \theta_2)$

$s_{12} = \sin(\theta_1 + \theta_2)$

$c_i = \cos(\theta_i)$

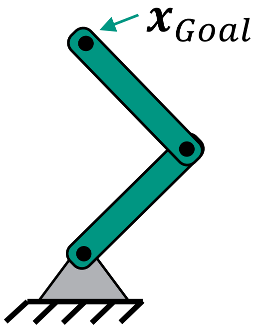
$s_i = \sin(\theta_i)$

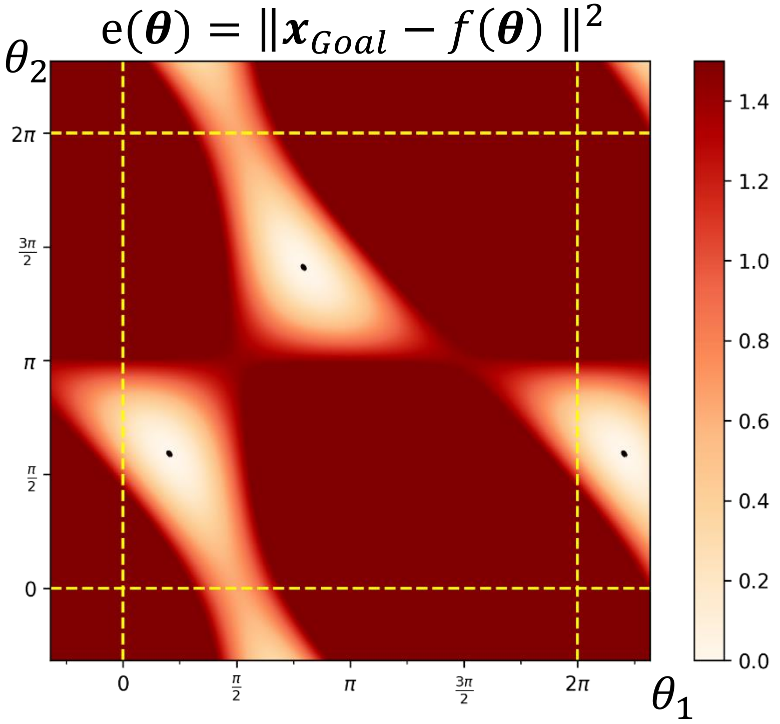
- For $\theta_2 = n \cdot \pi, \; n \in \mathbb{Z} \; J_f(\boldsymbol{\theta})$ is singular!

Pseudoinverse: Numerical Example (1)

2-DoF planar robot: $\boldsymbol{\theta} = (\theta_1, \theta_2) \in C$

Target pose $\boldsymbol{x}_{Goal} = (0, 1.2)^T$





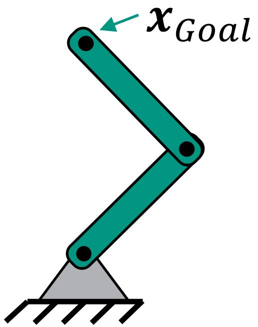
Pseudoinverse: Numerical Example (2)

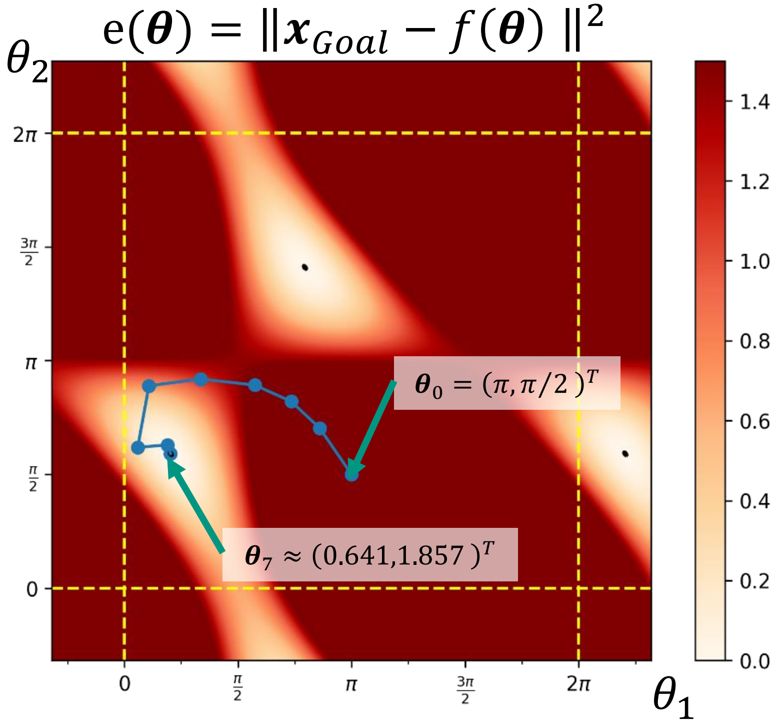
2-DoF planar robot: $\boldsymbol{\theta} = (\theta_1, \theta_2) \in C$

Target pose $\boldsymbol{x}_{Goal} = (0, 1.2)^T$

Step length: $\gamma = 0.2$

Start: $\boldsymbol{\theta}_0 = \left(\pi, \frac{\pi}{2}\right)^T$

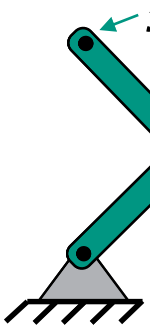




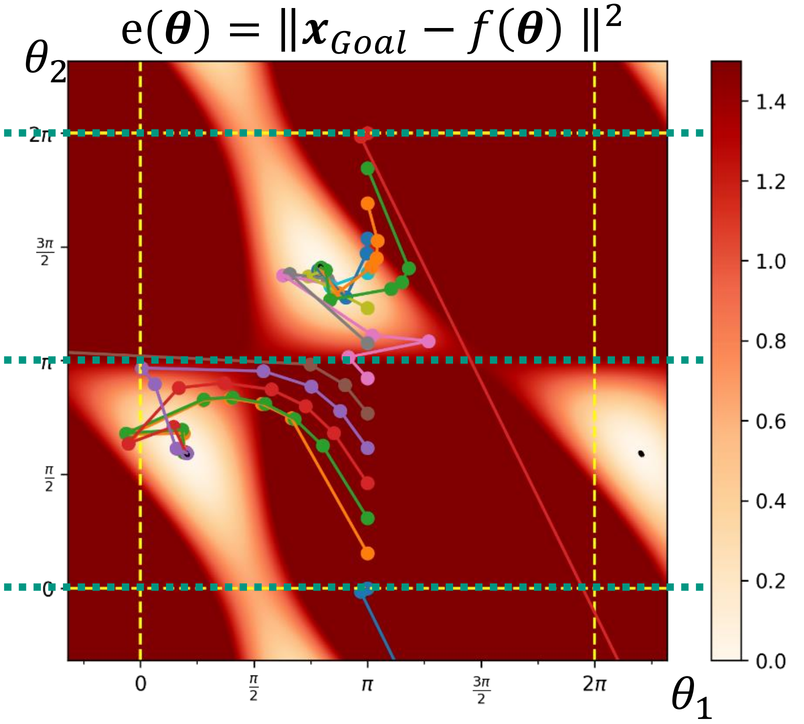
Pseudoinverse: Numerical Example (3)

2-DoF planar robot: $\boldsymbol{\theta} = (\theta_1, \theta_2) \in C$
Target pose $\boldsymbol{x}_{Goal} = (0, 1.2)^T$

Step length: $\gamma = 0.2$
Different starting points:
$\boldsymbol{\theta}_0 = \left(\pi, \theta_{2,start}\right)^T$



Singularities at $\theta_2 = n \cdot \pi$



Pseudoinverse: Singularities

- **Pseudoinverse is unstable in the vicinity of singularities**

- Ho to deal with singularities
  - Avoidance of singularities (not always possible)

  - **Damped least squares**
    (also **Levenberg-Marquardt Minimization**)

Pseudoinverse: Damped Least Squares (1)

- The pseudoinverse $J_f^+(\boldsymbol{\theta})$ optimally solves the equation $J_f(\boldsymbol{\theta})\Delta\boldsymbol{\theta} = \Delta\boldsymbol{x}$ for $\Delta\boldsymbol{\theta}$.
- Optimal refers to the sum of the error squares

$$\min_{\Delta\boldsymbol{\theta}} \left\| J_f(\boldsymbol{\theta})\Delta\boldsymbol{\theta} - \Delta\boldsymbol{x} \right\|^2$$

- **Approach:** Minimize instead (introduce regularization)

$$\boxed{\min_{\Delta\boldsymbol{\theta}} \left\| J_f(\boldsymbol{\theta})\Delta\boldsymbol{\theta} - \Delta\boldsymbol{x} \right\|^2 + \lambda^2 \|\Delta\boldsymbol{\theta}\|^2}$$

with a damping constant $\lambda > 0$.

Pseudoinverse: Damped Least Squares (2)

Approach:

$$\min_{\Delta\boldsymbol{\theta}} \left\| J_f(\boldsymbol{\theta})\Delta\boldsymbol{\theta} - \Delta\boldsymbol{x} \right\|^2 + \lambda^2 \|\Delta\boldsymbol{\theta}\|^2$$

This can be written as

$$(J^T J + \lambda^2 I)\Delta\boldsymbol{\theta} = J^T \Delta\boldsymbol{x}$$

This results in

$$\Delta\theta = \underbrace{(J^T J + \lambda^2 I)}_{\in\mathbb{R}^{n\times n}}{}^{-1} J^T \Delta\boldsymbol{x} = J^T \underbrace{(J J^T + \lambda^2 I)}_{\in\mathbb{R}^{m\times m}}{}^{-1} \Delta\boldsymbol{x}$$

Here: $J = J_f(\boldsymbol{\theta}), m = 6$

Pseudoinverse: Damped Least Squares (3)

- **Solution:**

$$\Delta\theta = \underbrace{\left(J^T J + \lambda^2 I\right)^{-1}}_{\in \mathbb{R}^{n\times n}} J^T \Delta\boldsymbol{x} = J^T \underbrace{\left(J J^T + \lambda^2 I\right)^{-1}}_{\in \mathbb{R}^{m\times m}} \Delta\boldsymbol{x}$$

- The damping constant λ>0 must be chosen carefully to ensure numeric stability
  - **Large enough** for numerical stability near singularities
  - **Small enough** for a fast convergence rate

- **Here:** $J = J_f(\boldsymbol{\theta})$

Numerical Methods: Stability Analysis (1)

- Both approaches (**pseudoinverse** and **damped least squares**) can become unstable due to singularities.
- Stability can be analyzed using **singular value decomposition** (SVD)
- **Singular value decomposition:** A matrix $J \in \mathbb{R}^{m \times n}$ is represented by two orthogonal matrices $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ and a diagonal matrix $D \in \mathbb{R}^{m \times n}$, in the form
$$J = UDV^T$$
- Without loss of generality: **Singular values** σ_i on the diagonal of D are sorted
$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m \geq 0$$

Numerical Methods: Stability Analysis (2)

Singular value decomposition : $J = UDV^T$

The singular value decomposition of J **always exists** and allows the following representation of J

$$J = \sum_{i=1}^{m} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T \,,$$

$\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ are the columns of U and V, $r = rang\, J$.

The following applies to the pseudoinverse J^+ (due to the orthogonality of U and V):

$$J^+ = VD^+U^T = \sum_{i=1}^{r} \sigma_i^{-1} \boldsymbol{v}_i \boldsymbol{u}_i^T$$

Numerical Methods: Stability Analysis (3)

- Reminder **Damped Least Squares**: $\Delta\theta = J^T(JJ^T + \lambda^2 I)^{-1}\Delta\boldsymbol{x}$

- The following applies to the inner matrix (to be inverted):

$$JJ^T + \lambda^2 I = (UDV^T)(VD^TU^T) + \lambda^2 I = U(DD^T + \lambda^2 I)U^T$$

- $DD^T + \lambda^2 I$ is a **non-singular** diagonal matrix with the diagonal entries $\sigma_i^2 + \lambda^2$.
  Therefore, $(DD^T + \lambda^2 I)^{-1}$ is a diagonal matrix with the diagonal entries $\left(\sigma_i^2 + \lambda^2\right)^{-1}$

- It follows:

$$J^T(JJ^T + \lambda^2 I)^{-1} = (VD^T(DD^T + \lambda^2 I)^{-1}U^T = \sum_{i=1}^{r} \frac{\sigma_i}{\sigma_i^2 + \lambda^2}\boldsymbol{v}_i\boldsymbol{u}_i^T$$

Numerical Methods: Stability Analysis (4)

- Pseudoinverse:

$$J^{+} = \sum_{i=1}^{r} \underbrace{\frac{1}{\sigma_i} \boldsymbol{v}_i \boldsymbol{u}_i^T}_{\to\infty\ (\sigma_i \to 0)}$$

- Damped Least Squares:

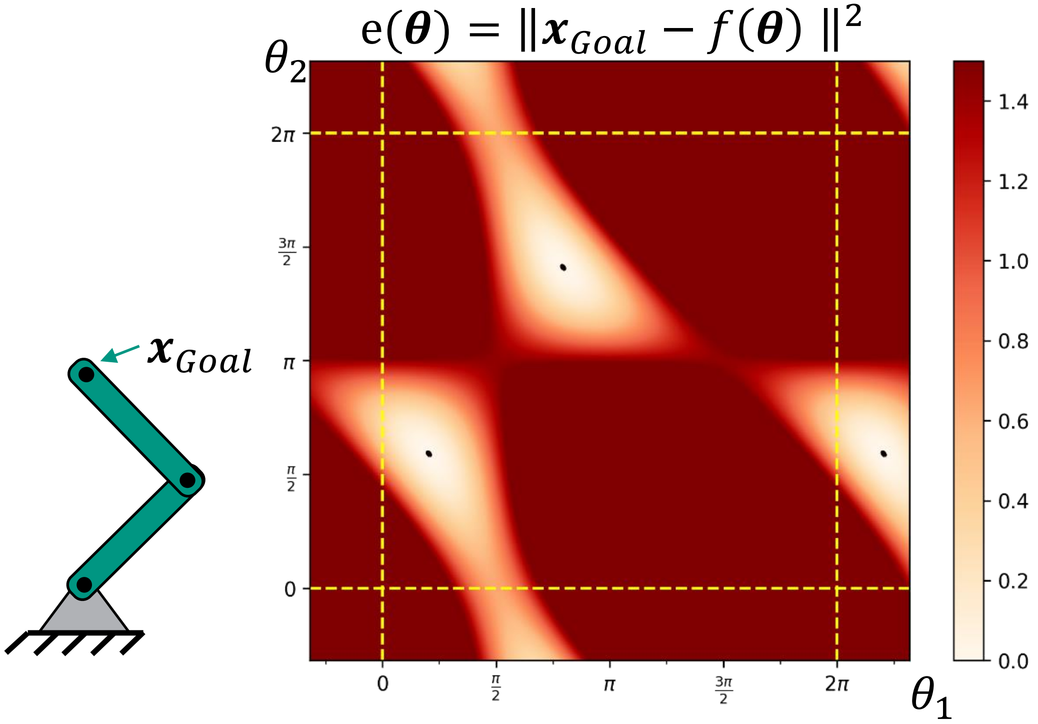
$$J^T (JJ^T + \lambda^2 I)^{-1} = \sum_{i=1}^{r} \underbrace{\frac{\sigma_i}{\sigma_i^2 + \lambda^2} \boldsymbol{v}_i \boldsymbol{u}_i^T}_{\to 0\ (\sigma_i \to 0)}$$

- The inversion of J has a similar form in both cases.
- The pseudo inverse becomes unstable when a $\sigma_i \to 0$ (singularity)
- For large σ_i (compared to λ), Damped Least Squares behaves like the pseudo inverse
- For $\sigma_i \to 0$, Damped Least Squares behaves well-defined

Damped Least Squares: Example (1)

2-DoF planar robot: $\boldsymbol{\theta} = (\theta_1, \theta_2) \in C$

Target pose $\boldsymbol{x}_{Goal} = (0, 1.2)^T$

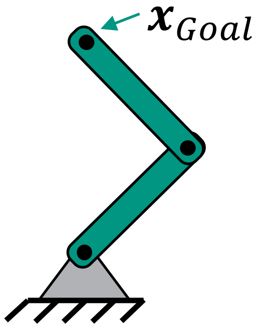


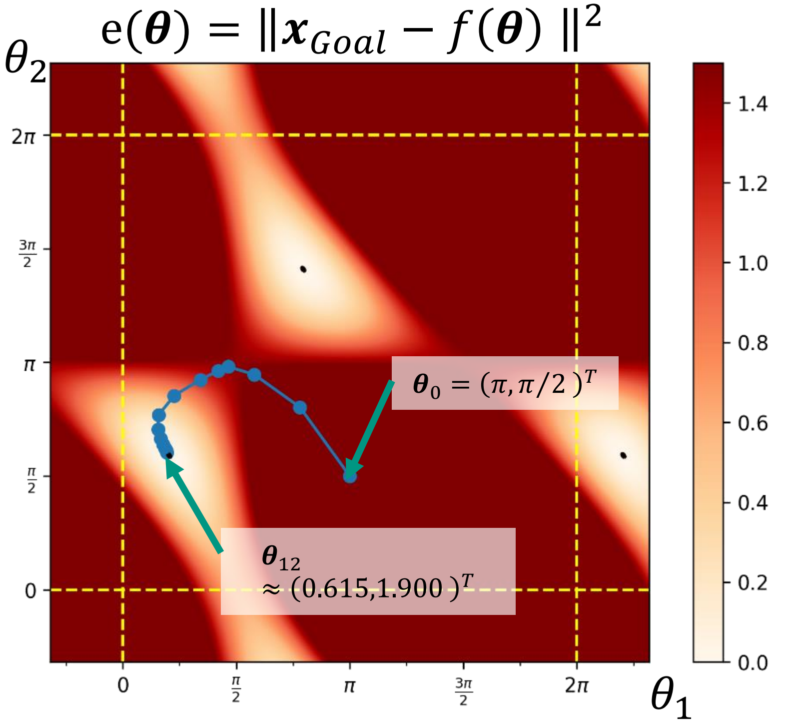
Damped Least Squares: Example (2)

2-DoF planar robot: $\boldsymbol{\theta} = (\theta_1, \theta_2) \in C$
Target pose $\boldsymbol{x}_{Goal} = (0, 1.2)^T$

Step length: $\gamma = 0.5$
Damping: $\lambda = 0.5$

Start: $\boldsymbol{\theta}_0 = \left(\pi, \frac{\pi}{2}\right)^T$

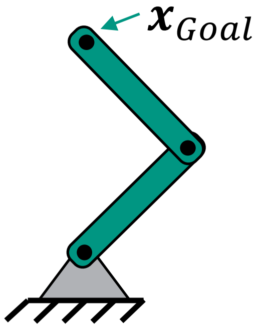


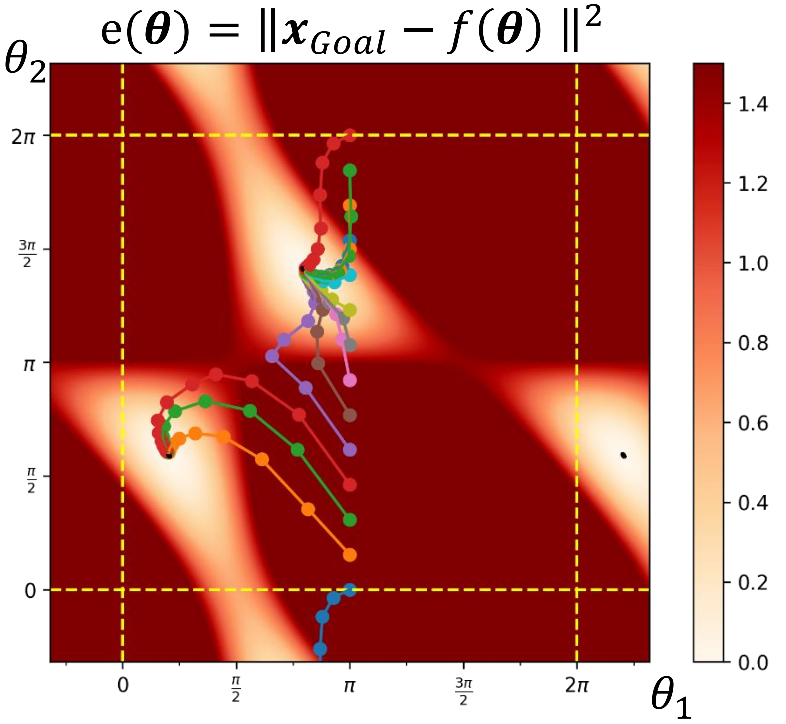


Damped Least Squares: Example (3)

- 2-DoF planar robot: $\boldsymbol{\theta} = (\theta_1, \theta_2) \in C$
- Target pose $\boldsymbol{x}_{Goal} = (0, 1.2)^T$

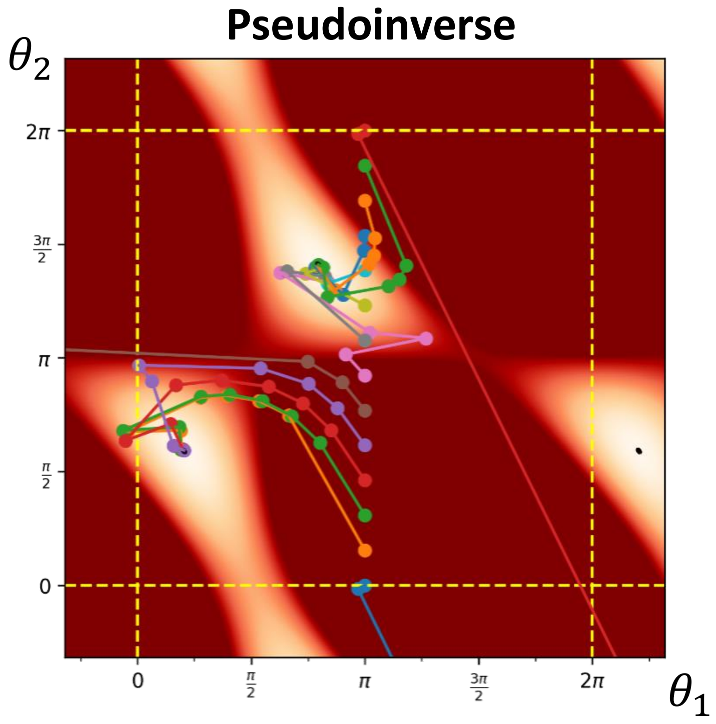
- Step length: $\gamma = 0.5$
- Damping: $\lambda = 0.5$
- Different starting points:
  $\boldsymbol{\theta}_0 = \left(\pi, \theta_{2,start}\right)^T$

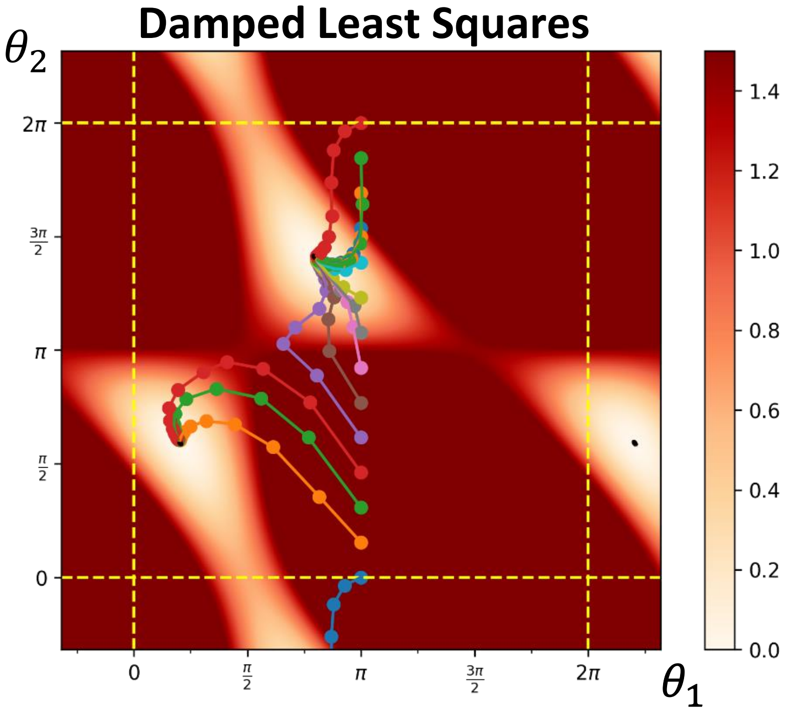




Comparison: Pseudoinverse vs. Damped Least Squares

Singularities at $\theta_2 = n \cdot \pi, n \in \mathbb{Z}$





Overview

Inverse kinematic problem

Closed-form methods

- Geometric
- Algebraic

Numerical methods

- Gradient descent
- Jacobian based and pseudoinverse based methods

Summary

Summary: Kinematics

Direct kinematics:

$$f\colon\ \mathbb{R}^n \to\ \mathbb{R}^m \qquad \boldsymbol{x} = f(\boldsymbol{\theta})$$

Inverse kinematics:

$$F\colon\ \mathbb{R}^m \to\ \mathbb{R}^n \qquad \boldsymbol{\theta} = F(\boldsymbol{x})$$

Cases:

- There is a unique solution.
- There is a finite number of solutions.
- There is an infinite number of solutions.
- No solution exists.

Important Spaces in Robotics